CS 251: Bitcoin and Crypto Currencies Fall 2015

Project 2

due: October 21, 11:59 PM via email to cs251.stanford.fall.2015@gmail.com

Introduction

In Lecture 7 we discussed a number of mining strategies. In this project you will implement three strategies discussed in class. The optional last exercise is open-ended where you can experiment with mining strategies of your own. If you wish, you can enter your miner in a class competition.

Getting started

1. Download the code from the course website and and import it into your favorite IDE. You can use maven to download the required dependencies.
2. Familiarize yourself with the starter code. You should especially look at the `Miner` interface. All your miners should implement this interface. Look at the JavaDoc that explains what each function does. Additionally we provide you with an implementation of a `CompliantMiner`
3. Your goal in all parts is to maximize your profit **relative** to the other miners.
4. The class `MiningSimulation` provides a set of tests that your Miners should pass. The revenue goals should be achievable with standard implementations but feel free to try to surpass them. Especially if you plan on entering the class competition.

Simulation Model

The simulation uses a simplified model of the real Bitcoin network world to simulate the mining and propagation of blocks. A fixed number of blocks will be simulated. You do not need to understand all the details of the simulation but the most important points are

1. The simulation goes in discrete iteration where each iteration consists of a mining round and propagation round
2. Each miner draws a creation time from $Exp[hashRate]$ and the miner with the lowest creation time mines a block. The probability of mining a block in each iteration for a miner i is, thus, $\frac{hashRate_i}{\sum_{j=1}^{n} hashRate_j}$.
3. With small probability a second block gets mined at the same time (by the miner who drew the second smallest number). Both blocks will be propagated at the same time. With even smaller probability a third block is mined and so on.
4. If a miner s wants to broadcast a block he draws transmission times for each other miner r from $Exp[connectivity_s * connectivity_r]$. If two miners s_1 and s_2 broadcast a block to r at the same time then the probability of s_1's block arriving first is $\frac{connectivity_{s_1}}{connectivity_{s_1}+connectivity_{s_2}}$

5. If a miner upon receiving a block wants to broadcast a new block he draws a set of transmission times for the other agents as in 4. but delayed by the current time. Consequently there exists a first mover advantage in block propagation.

Submission: For all exercises, submit the source code, especially for the newly created miners. As usual, please create a tar or zip file of your submission and email it to the address at the top of the page.

Exercises

1. Create a "majority miner" (by extending the `Miner` interface) that performs a 51% attack if it is capable. A 51% in this context means extracting as much relative profit as possible. The network may have some natural churn, so your status as a majority miner may be changing.

2. Create a "selfish miner" that performs a selfish mining attack if profitable. This strategy is outlined in Chapter 5 (Section 5.5) of the NBFMG textbook.

3. Create a "whale miner" that can profit by forking to steal "whale transactions" when profitable. That is, when an unusually large transaction is mined by a competitor, your miner should temporarily reject that block and try to re-mine a longer fork where they keep the large transaction fee for themselves.

4. (Optional) Create the most powerful miner you can think of. It should be able to intelligently handle a as many different scenarios as possible. We will test in a variety of settings (to be kept secret) as well as against your fellow students' miners in a tournament. To enter your miner in a class competition, please name the miner `<YourLastname>Miner`.